# Informe Técnico y Recomendaciones

Este informe detalla el estado actual de la plataforma a nivel técnico y propone mejoras para asegurar su escalabilidad y mantenibilidad.

## Estado Actual

- **Stack Tecnológico:** Moderno y potente. Next.js 16 (Release Candidate), React 19, Prisma, TailwindCSS.
- **Arquitectura:** Clara separación de responsabilidades (Admin vs Client). Diseño Multi-tenant correcto.
- **Base de Datos:** PostgreSQL con un esquema relacional robusto.

## Hallazgos y Riesgos Detectados

### 1. Seguridad y Consultas SQL (Criticidad: Alta)

Se ha detectado el uso de consultas SQL crudas mediante `sql.unsafe` en rutas críticas (ej: `app/api/cases/route.ts`).

- **Riesgo:** Aunque actualmente se usan parámetros manuales ( `$1` , `$2` ), este patrón es frágil. Un error futuro en la concatenación de strings podría abrir brechas de **SQL Injection**.
- **Recomendación:** Migrar estas consultas a métodos nativos de **Prisma ORM** (ej: `prisma.case.findMany` ) o usar un "Query Builder" seguro. Evitar `sql.unsafe` a toda costa.

### 2. Escalabilidad y Rendimiento (Criticidad: Media)

- **Paginación:** La API de listado de casos ( `GET /api/cases` ) hace un `SELECT *` sin límites ( `LIMIT/OFFSET` ).
- **Riesgo:** Si un despacho acumula cientos de expedientes, esta llamada se volverá lenta y podría colapsar el navegador del cliente.
- **Recomendación:** Implementar paginación en el backend (ej: recibir `page` y `limit` en la query).

### 3. Estabilidad de Versiones (Criticidad: Media)

- **React 19 RC:** El proyecto usa una versión candidata (RC) de React 19 y Next.js 16.
- **Recomendación:** Vigilar de cerca las actualizaciones estables. Las versiones RC pueden tener bugs no documentados o cambios que rompan la app ("breaking changes").

### 4. Tests (Criticidad: Media)

- No se observa una estrategia de testing automatizado (Unitarios o E2E).
- **Recomendación:** Implementar al menos tests básicos para flujos críticos:
  - Creación de expedientes.
  - Login y permisos (asegurar que un cliente no vea datos de otro).

### 5. Infraestructura de Pagos (Criticidad: Alta)

- **Falta de Pasarela:** No se ha implementado la integración con pasarelas de pago (ej: Stripe, Redsys) para gestionar las suscripciones de los despachos (SaaS).
- **Riesgo:** Imposibilidad de monetizar la plataforma de forma automatizada. Requiere gestión manual de cobros.
- **Recomendación:** Integrar una solución como **Stripe Billing** para gestionar planes, ciclos de facturación y pagos recurrentes.

## Plan de Acción Recomendado

1. **Refactorización de Consultas:** Reemplazar `sql.unsafe` por Prisma Client en los endpoints principales.
2. **Añadir Paginación:** Modificar `GET /api/cases` para devolver resultados paginados.
3. **Auditoría de Seguridad:** Revisar todos los endpoints que reciben parámetros del usuario.
4. **Setup de Tests:** Configurar Jest o Vitest e implementar 2-3 tests de integración clave.